VOL. 10, NO. 3, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

SOFTWARE DEFINED NETWORK AND OPENFLOW: A CRITICAL REVIEW

Shivaleela Arlimatti, Suhaidi Hassan, Adib Habbal and Suki Arif
InterNetWorks Research Lab, School of Computing. Universiti Utara Malaysia, UUM Sintok, Kedah, Malaysia
E-Mail: sarlimatti@gmail.com

ABSTRACT

Software Defined Networks (SDN) is an emerging new network paradigm which enables network programmability and breaks the network vertical integration by separating network intelligence from underlying network devices such as routers and switches. SDN promotes the logically centralized control to program the network. SDN decouples data plane and control plane of the network devices to simplify the network management and great innovation by network programmability, using OpenFlow as a communication protocol between SDN controller and network elements. This paper presents a comprehensive critical survey on SDN and OpenFlow. The main aim of this paper is to give a brief introduction of SDN, the basic architecture of SDN and to show the control plane and data plane separation. The building blocks of SDN as layers are provided with study of infrastructure, southbound, controllers, northbound and network applications. Later research challenges and distributed computing in SDN are discussed to provide future researcher's brief idea about the future scope in the field.

Keywords: software defined networks, open flow, controllers, distributed computing.

INTRODUCTION

SDN is an emerging network approach to design the network with software programs. This programmability and automation greatly reduces the time spent by staff provisioning and maintaining the network device-by-device, effectively increasing network agility. SDN is a programmable network, where behavior of network elements and their flow of control to enable the implementation of SDN concepts in both hardware and software. These software programs operate independently of network hardware. SDN uses Open flow(McKeown et al., 2008) as a communication protocol between SDN controller and network elements. The controller is used as a center of programmable control. Having centralized control makes a whole network run efficiently and effectively. SDN and Open Flow simplifies network management by decoupling the control plane and data plane of the network(Kreutz, Ramos, & Verissimo, 2013)

SDN is a “programmable network”, proposed to facilitate evolution of networks and is a paradigm where in control decisions decouple from forwarding hardware. The network intelligence in SDN is logically centralized and called as control plane and forwarding devices are programmed to forward packets through open interface and is known as data plane. Recently SDN is growing with very fast pace and there are many research challenges which have to be addressed.

In SDN, a centralized single controller managing all the forwarding devices inside the network may show Singled Point Of Failure (SPOF) and some times the network is overloaded due to heavy traffic which leads to scaling limitations. One controller is not enough to manage any data center network with large number of forwarding devices. A centralized cluster node system offers high throughput inside the data center network.

The main inspiration of SDN is to make use of network devices with standard application program interface (APIs) to allow 3rd party operators to control the data flow through the network. SDN reduces complexity of networks by providing up to date network view to the programmers. This signifies the development, management applications are simplified. Control layer of SDN handles tedious work like synchronization and distribution.

The main objective of this paper is to provide a critical survey on SDNs with Open Flow. Distributed computing has a common goal to solve computational problems by sharing the existing resource in heavy workload. Each controller will have its own task and, coordinates with other controllers to provide the desired service. A distributed computing system scales up to meet the requirements of any environment. These controllers may be centralized cluster nodes or physically distributed elements. Distributed controllers have a property of fault tolerance, when any node fails in the network; the adjacent node will take the in-charge of the failed node. Even controllers will have a property of tolerating crash failure.

This paper is organized as follows. In the next section SDN architecture is discussed. Later some issues on SDN layers are explained briefly. Data center network and Research challenges are discussed in the subsequent sections.

SOFTWARE DEFINED NETWORK ARCHITECTURE

SDN is a new emerging architecture in which control plane and data plane are decoupled and controllers are programmable. In this architecture network intelligence is centralized which controls the whole network. The network appears as a single logical switch to the applications. By the centralization of control plane,

VOL. 10, NO. 3, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

network admins are getting flexibility to manage, configure and optimize network resources through SDN programs(Beheshti & Zhang, 2012**)**. SDN architecture supports APIs that implement common network services, which include access control, multicast, routing, bandwidth management, storage optimization, traffic engineering, energy usage and different types of policy management. The SDN architecture defines consistent policies on both wireless and wired connections. It supports network virtualization to enable scalability in the data center, bandwidth optimization and server utilization.

SDN is developed to enable simple and programmable control to manage and control data path in forwarding devices. The SDN architecture is flexible and control plane handles the forwarding of packets through switches.

When the first packet enters the network, the network element, switch forwards this packet to SDN controller, then the controller takes a decision to add the flow in the forwarding table or drop the packet for security purpose. As per the controllers dictation switch forwards the packet to its appropriate destination.

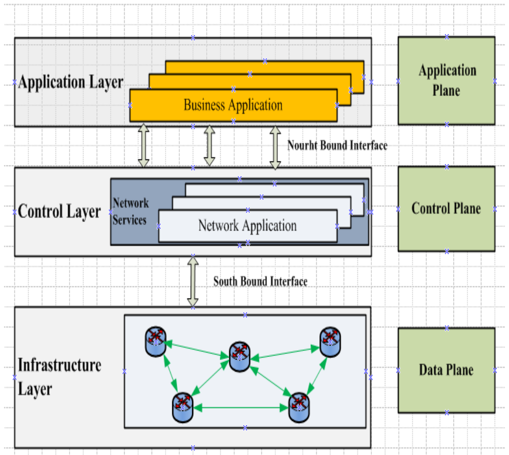

Figure-1. Software defined networking architecture.

According to Open Networking Foundation (ONF)(Wasserman & Hartman, 2013) SDN architecture consists of three planes, as shown in Figure-1, where the first plane is Application plane, Second plane is control plane and third plane is Infrastructure plane or forwarding plane. In the network all the planes are mapped with interfaces to communicate with other planes.

Application plane is the top most layer of the SDN architecture. It contains the applications like access control, security monitoring, and energy efficient networking and so on. The application layer and control layer are connected through an API known as north bound API. The main objectives of the north bound interface are loop avoidance, security, routing, path computing, management and policy requirements. In SDN, information flows from the SDN controller to SDN application which is used for services or for the applications. SDN application controls the SDN orchestrator by modifying the routing within the network(Gurbani, Scharf, Lakshman, Hilt, & Marocco, 2012).

Control plane contains the software-based SDN controllers. These controllers control all the network devices which are presented in the infrastructure layer. This controlling mechanism is handled with Open Flow protocols and open flow switches. The control layer is connected to the infrastructure layer by an application program interface (API). This API is known as southbound application program interface. This gives the communication between the controller and the network devices. The actual management and configuration within the network is carried out through the south bound interface.

Infrastructure plane contains all the devices of the network like switches, routers and so on. This layer is nothing but the data plane or the forwarding plane of the network. These network devices are programmed and controlled by a controller with Open Flow protocol.

SOFTWARE DEFINED NETWORKING LAYER

SDN uses bottom up approach with five main layers as shown in Figure-2. Each layer has its own functionality. The following sub section explains each layer based on different technologies, concepts and properties.

Network infrastructure layer

Network infrastructure involves different network elements such as switches and routers. These are known as forwarding devices. SDN represents these forwarding hardware elements accessed by an open interface by control logic. Always SDN considers switches as simple forwarding devices. Nowadays virtualized networks and data center infrastructure are using these software switches and are emerging as a promising solution to data center networks because data center contains more virtual access ports than its physical access ports(Casado, 2013). The ancient technology, Network Virtualization is the driving force for this technology.

The data plane elements have specialized in data forwarding based on forwarding rules or flow rules. Flow rule or forwarding rule is a combination of more than one matching fields. We can divide these rules into two categories; processing forwarding rules and installing forwarding rules**.**

Inside the forwarding elements, flow tables are defined sequentially. These flow tables will have flow rules, which tell how the packet is handled. An Open Flow forwarding switch is based on flow tables, each entry in the flow table consists of three categories(Kreutz et al., 2014).

- A matching rules
- Set of instructions
- Statistics (statistics of matching packet)

When a packet enters the data plane element, then lookup process is carried out to find the match. If no match found, then this will contain a default rule. This rule tells the forwarding element to send that packet to the controller to set new rules. The majority of the rules follow table sequence number and row order in the table. Forwarding device is flexible to take any action on the arrived packet. Some of the example actions are to forward packet to port, forward packet to controller, drop packet and send packet to the next table.

Southbound interface

South bound application program interface is the link between controller and forwarding devices. Forwarding devices are controlled by application program interface (open interface). Open Flow protocol is the best example for south bound interface, it shows execution of the switch to controller communication and vice versa(McKeown et al., 2008).

Open Flow is widely deployed and accepted open southbound interface standard. The Open Flow provides information sources to controllers such as, when port change or link is triggered messages are sent by forwarding elements to the controller, forwarding elements generate flow statistics and controllers collect. When the new incoming packet having action “send to controller” in matching entry of the forwarding table, then forwarding devices send this packet to controllers. This gives interoperability between Open Flow equipments to different vendors.

Open Flow defines set of instructions exchanged between the forwarding device and a controller through secure channel. By this remote controller can update, delete or add flow entries from flow tables. This can be done in two ways Reactively and Proactively. Reactive means each time a decision is made, forwarding element should consult to controller. In a proactive way controller push policies to switches.

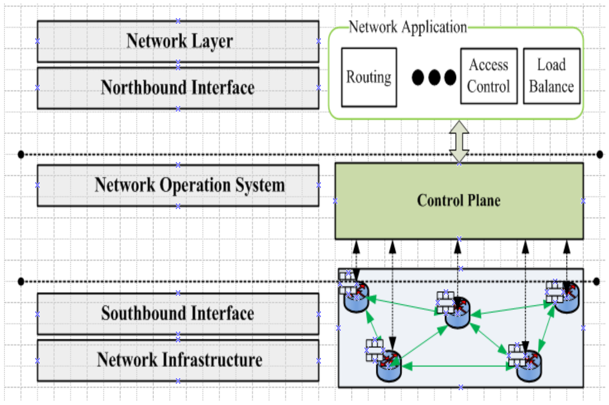

Figure-2. SDN layers.

Network operating system/ control layer

In a decoupled system (data plane and control plane), control layer supports for the programmatic interface inside the network. On centralized control plane applications are written to see the network as a single element. The controller communicates with forwarding elements through Open Flow protocol.

SDN promises to manage the network by logically centralized controller, which is provided by Network Operating System (NOS) to ease the burden of network problem. NOS provides services such as network topology information and network state generic functionality, distribution of network configuration and device discovery.

There are three layers of the control platform

- The application, Orchestration and service
- Core controller functions
- South bound communication elements

The controller should provide a base network service function, like base service operating system, input, output operations control, program execution, protection and communication. Other operating service uses these services.

Network applications use control functionalities management, security mechanism and shortest path forward. Then the notification manager will process, forward events and receive.

Northbound interface

A northbound interface allows any lower level component to communicate with a higher level component. This is the communication interface between the application layer and the control layer in the architecture of SDN. In data center network northbound application program interface manages orchestration and automation, and actively shares data between the systems.

North bound application program interface can support for network functions like loop avoidance, security, routing, load balancing, computation and many other network functions.

Network application layer

Management applications execute the control logic to install commands in the data plane and dictate the operations of the forwarding elements. SDN is deployed in the data center for management applications to perform network functions such as load balancing, routing, reducing power consumption and security enforcement. The main goal of management application is to maximize network utilization, optimization of load balancing, engineer traffic to minimize power consumption and optimization of traffic.

Load balancing is the one of the goals of the SDN/Open Flow, it emerged for increasing scalability of the network. When a new server is installed in the network, Load balancing will take automatically by distributing the traffic, network load and computing capacity. This provides flexibility to network and it will simplify network management.

Traffic optimization in large service providers for dynamic scaling out and some other applications are video streaming(Jarschel, Wamser, Hohn, Zinner, & Tran-Gia, 2013)and multiple packet scheduling to improve Quality of Services(Ishimori, Farias, Cerqueira, & Abelém, 2013)**.**

SDN FORDATA CENTER NETWORK

The data center is a place or it is the area of resources like storage, computation and network, which are implemented by a communication network. Data center network is the interconnection of all data center resources.

Nowadays companies build their own data centers to minimize operational expense. Administrator of the company always invest expensive servers instead of low cost commodity hardware because data center requires thousands or more commodity servers for high performance demand. Data center network is always shared by more than one tenant. It has two main challenges,

1. **Topology and routing:** Traditional network topology follows a hierarchical method which requires high performance hardware. There is a lack of powerful devices, which leads to failing of adequate capacity in data centers. But data center followsthe parallel paths for high performance demands and fault tolerance.At thesame time it is necessary for balancing the traffic on parallel links for total utilization of the network with limited capacity of switch forwarding tables.

2. **Multi tenancy:** To reduce the economic scale, data center networks place a different set of tenants on the shared network, minimizes the infrastructure investments. But it is necessary for the data center administrator to distinguish their tenants from traffic individually. As the tenant's traffic increases, switches require multiple flow table entries to represent tenant's tasks.

By the above limitations data center network requires controller platform for automation of scalable resources.

Recently all the IT centers and the services are dependent on highly efficient and scalable data centers. Still, there are challenges in this area regarding storage, network king, computing, flexibility, resilience, resource utilization, latency, agility to provide a network resource (e.g. Network function storage) and orchestration in computing (Kant, 2009)**,** (Bari et al., 2013**).**

Evolution of SDN meets the higher and challenge demand of data centers. Due to the dynamic complexity adapting to application needs, data center networks provide rapid service for higher workloads in less capacity. Data center network with SDN can solve many problems, such as improved network management(Arefin, Singh, Jiang, Zhang, & Lumezanu, 2013), network utilization optimization (Raghavendra, Lobo, & Lee, 2012), network migration(Blenk & Kellerer, 2013).

SDN data centers satisfy high traffic conditions and switches are turned off when they are not in use. This saves the energy 25-62% with changing traffic condition. Virtualization of network functions increases further energy savings.

In 2012, Google represented the real application and need of SDN in data centers, presented at Open Networking Summit(Foster et al., 2013)implementation of SDN network connections in data centers.

Now there is a need for traffic engineering, custom routing and level of scalability in data centers. Still, there exist the challenge like bottleneck problem in between data plane and control plane communication. These are the some of the issues which should be considered as future research by the authors.

SDN allows infrastructure providers custom addressing, middle box placements, isolation of virtual networks and virtual cloud applications to expose networking functions to its customers (Benson, Akella, Shaikh, & Sahu, 2011)**,(**Calyam et al., 2013). The SDN detects abnormal behavior of network operation in the data center. By using different behavioral model and gathering the required information from elements of data center like infrastructure elements, operations and application to capture the control traffic.

CENTRALIZED VS DISTRIBUTED CONTROLLERS

SDN specifies switch is always controlled by a controller, in this manner it is not necessary that the controller must be centralized; some time it may be distributed controller. Open Flow defines only controller to switch communication or vice versa. But controller to controller communication is not defined by Open Flow. The Internet Engineering Task Force(IETF) is currently working on inter controller communication and developing a protocol SDNi (Stallings, 2013),to interface different domain controllers of SDN. SDNi plans to elaborate on maintaining flow setup time originated by information like service level agreements and path requirements applications Reach ability information exchange to support inter domain routing.

We can further define SDN as the separation control plane and data plane and control intelligence is removed from network devices, which will be a simple packet forwarding device. This Open Flow based SDN architecture Figure-3 shows the master client architecture, which will not be suitable for large data center networks. A single, centralized controller that cannot manage large data center network. It represents Single Point Of Failure (SPOF) and have scaling problem. Therefore SDN/Open Flow supports for multiple controllers, to handle large networks.

There are two types of organizations for distributed controllers.

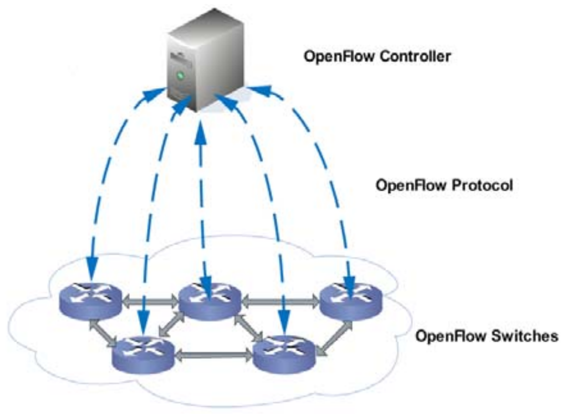

Figure-3. Single controller SDN.

- Hierarchical organization of controllers(Hassas Yeganeh & Ganjali, 2012) .
- Flat organization of controllers(Tootoonchian & Ganjali, 2010).

These two types of distributed controllers are designed for large data centers to share huge workload and information among themselves.

Hierarchical organization of controllers

In hierarchical organization, there is no interconnection between the individual local controllers in the first layer of the architecture. Bottom layer takes care of local control applications and all these local controllers are handled by main controller at the top level.

Flat organization of controllers

In flat organization, network is partitioned into several smaller domains and each domain is controlled and managed by individual controllers. And each individual controller is interconnected with other controller by an interface to exchange the routing information.

(Phemius, Bouet, & Leguay, 2013)worked on multi domain networks. They placed to interconnect enterprise network, data center network, and mobile network and customer sites. They divided the network into domains, interconnected with network technologies with high capacity lines to low bandwidth links, costly and highly secured links to unsecured and cheaper ones. They're distributed multi domain network controllers are robust to failure. This gives motivation to think about the communication cost of the network as a whole, either inter-communication cost or it may be intra-communication cost.

Hyper Flow(Mendonca, Nunes, Nguyen, Obraczka, & Turletti, 2013)(Tootoonchian & Ganjali, 2010), and Onix(Koponen et al., 2010)are physically distributed controllers but are maintained logically by centra controllers, butr. This reduces the communication overhead within local controllers. Kandoo, used the idea of hybrid approach for design, local controllers will maintain local applications and global controller will maintain all the local controller, acts like centralized controller. This minimizes the load of the global controller. Flow Visor, proxy controller, adds network virtualization, and allows multiple controllers concurrently control the set of switches. A distributed control plane can be group of control nodes in one place or physically distributed control nodes. Cluster of control nodes is used in data centers to get high throughput. Physically distributed control plane nodes are used in Wide Area Network (WAN) to interconnect data centers. There may be with group of controllers inside data centers.

Distributed controllers have a property of fault tolerance. It's natural in any network, node failure occurs, and then adjacent node takes over the charge of the failed node and does not disturb the network application. The same is continued for controller failure, fault tolerance in controller crash failure. Multiple controllers are used to minimize the latency and maximize the fault tolerance.

(Heller,2012), focused on determining the number of controllers are needed in the network and location of the controller in the network topology, by choosing the optimization of worst case and average case latency. Due to, network conditions change with respect to time, fixed location of controllers may not be a solution forever. (Bari et al., 2013) proposed dynamic deployment of controllers within Wide Area Networks (WAN). They considered a number of controllers and the location of the controllers to place the controller dynamically, by reducing flow setup time and communication overhead.

There are three main reasons to go for a distributed control plane.

Scalability: A Single controller can feasible to manage a limited number of network devices due to its limited capacity. So there is a need to distribute the control plane to manage large networks.
Privacy: Network administrator may select different privacy policies in logically different controller or domain.
Interconnected deployment: A data center network may contain both SDN and Non SDN network infrastructure. So dividing the network into individually manageable domains allow for network flexibility.

DISTRIBUTED COMPUTING IN SDN

Distributed computing is an area in Computer Science, in which computational problems are solved by distributing the problem into many modules; each module is solved by one or more resource.

In distributed computing, the networked computer components communicate and coordinate with each other by passing messages. When a program needs to cooperate with each other they coordinate by exchanging the information among them. The main motivation of using distributed computing in SDN is to share the existing resources and workload. The controllers communicate with others to achieve a common goal. Same time each controller will have its own task, but coordinating with other controller to provide communication service.

The main property of distributed computing is local memory and message passing among themselves. All the autonomous computational systems, controllers will have their own local memory. An autonomous system communicates with each other by passing messages. The main challenge of distributed computing in SDN is, controlling the cost of resource such as computing cost, storage cost and network. As the member of forwarding element increase controller should support to extend the system by increasing the number of controller at reasonable cost.

The main characteristics of distributed computing in SDN are as follows,

- Concurrency of controllers
- Lack of a global clock
- Independent failure of controllers

Any controller or network forwarding device may fail independently any time. The controller hides the faults from application programs to complete the given task in spite of failure of forwarding elements. As a whole distributed computing will improve the scalability and resilience of the control plane.

RESEARCHTRENDS

There are still many research problems to make SDN optimize across the network and more efficient. In the following subsections research trends are briefed according to SDN layers.

Infrastructure layer

Open Flow matching rules are installed in flow tables inside switches. These Open Flow rules are very complex compared to the forwarding rules in traditional network devices. These flow rules consist of three components

- Flow matching rules: Flow matching rules, are used for matching incoming packets.
- Counters: Counters, are used for collecting statistics for one particular flow. It may be duration of flow or the number of bytes.
- Actions: Actions or Set of Instructions, take the actions on matching packets, such as forward the packet to the next device normal procedure for forwarding, or if the match is not found in the matching table is sent the packet to controller to take further action, or dropping the packet according to the instruction given by the controller.

One of the challenges is to provide flow tables with high capacity and storage to store the flow rules(Appelman & de Boer, 2012). TCAMs are used to store the flow tables, but are expensive and more power consuming. This represents more power consumption in switch(Kannan & Banerjee, 2013).

The limiting factor that shall be addressed during switch design is to provide clean feature planning during the design process of the switch because; the throughput of the Open Flow switch is from 38-1000 flow mod/Sec. The achievable throughput is less than 500 flow mod/Sec(Stephens, Cox, Felter, Dixon, & Carter, 2012).

Control layer

Controllers are the backbone of the SDN architecture. Nowadays, research work is carried out to increase the scalability, modularity and operator friendly software. Controllers are distributed to share the workload among themselves. These distributed controllers face many challenges such as fault tolerance, consistency, synchronization, load balancing and latency between controller and forwarding elements(Schmid & Suomela, 2013)(Berde et al., 2014)(Koponen et al., 2010).

A controller communicates with its neighbor's controller by east/westbound API's. (Shin, Nam, & Kim, 2012). These are used in hierarchical designs of the controller in inter and intra data center networking. This enables to increase modularity and scalability of the controller in hierarchical design of the controller. It is very important to have interoperability between all the controllers, east/westbound API's between controllers e.g. SDN (Yin et al., 2012). Research is going on in this field very rapid. Low latency and high availability of controllers work well in small networks. When it comes to large networks, distributed controllers will take care of high availability of controllers by distributing the workload among the controllers. In interconnection of large network controller location plays an important role(Levin, Wundsam, Heller, Handigol, & Feldmann, 2012). This high availability is achieved by improving southbound API's and by placing controller heuristically. The only goal is to increase the scalability to accommodate more forwarding devices to connect to many controllers with cost efficient and cost effective way(Daniel Philip & Gourhant, 2014).

Application layer

Management system externally extracts information about network devices and network functions. We can get lots of work in controller switch interaction or communication by southbound API. There is still very less work in Application layer. Like southbound, controller and application layer communication is through northbound API. So researchers will have lot of scope in between the application layer and control layer communication, with northbound API.

SDN decouples control plane and data plane. Some network configurations are reactive where, the first packet of the network is sent to the controllers by forwarding elements. The main aim of applications is to build traffic by optimizing load balance, minimizing power consumption, maximum network utilization and traffic optimizing concept. Load balance is envisioned application for SDN/Open Flow(Wang, Butnariu, & Rexford, 2011), (Handigol et al., 2011), (Handigol, Seetharaman, Flajslik, McKeown, & Johari, 2009). Researchers (Wang, Butnariu, & Rexford, 2011) provided wildcard–based rules for proactive balancing of load. These wildcard is utilizing client requests on the basis of IP prefixes, allows directing and distributing group of requests without interference of the controller. The flow requests or the controller applications are required to manage and monitor network traffic, especially in the case of bottleneck.

In data center network topology, to avoid bottlenecks from incoming packets to compute the path Linear Bisection bandwidth technique is adopted(Al-Fares, Radhakrishnan, Raghavan, Huang, & Vahdat, 2010). Similarly, one more application is, traffic optimization for dynamic scaling the network. Monitoring the network by improving the Open Flow features reduce the control plane's overload with respect to data plane statistics.

Cloud computing is globally distributed for its services and experiences. Dynamically it manages and optimizes computing infrastructure. Cloud supports for distributed computing. Managing distributed application is really hard. To handle this task Open Application Delivery networking is proposed(Paul & Jain, 2012). In which application service provider (ASP) is allowed to express and enforce traffic management and application delivering policies whenever required.

To get the performance required within the multi-domain heterogeneous networks like interconnected data center networks, social networks, enterprise networks, programmability of the network is necessary. By doing this some of the queries are solved within the nodes, which reduce the communication between the controllers. This helps to get full scalability in multi domain networks.

CONCLUSIONS

SDN has been the new network paradigm in distributed computing in recent years. To enable SDN to be adopted fully, several issues have to be addressed comprehensively. In this paper, critical review has been carried out on SDN and Open Flow. The basic architecture of the SDN is given and layers of the SDN are discussed to highlight some open issues to readers. The distributed computing paradigm has been explored to avoid the controller failure, such as SPOF and for heavy workload distribution. The contribution of the paper can be categorized into; reviewing emerging SDN technology with existing traditional technology. Current challenges of the SDN are introduced and discussed. The paper concludes by exposing the possibility of employing SDN framework in data centers. For future works, it is proposed that SDN technology can is combined with NFV for virtual use of the network to reduce the investment cost in hardware appliance.

ACKNOWLEDGEMENTS

This research is one of the Malaysian Government funded projects under Ministry of Higher Education (MOHE) Long-Term Research Grant (LRGS) with reference number LRGS/TD/2011/UKM/ICT/02.

REFERENCES

Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., & Vahdat, A. (2010). Hedera: Dynamic Flow Scheduling for Data Center Networks. Paper presented at the NSDI.

Appelman, M., & de Boer, M. (2012). Performance analysis of openflow hardware. Master's thesis, University of Amsterdam, February, 11-20.

Arefin, A., Singh, V. K., Jiang, G., Zhang, Y., & Lumezanu, C. (2013). Diagnosing Data Center Behavior Flow by Flow. Paper presented at the Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on.

Bari, M. F., Boutaba, R., Esteves, R., Granville, L. Z., Podlesny, M., Rabbani, M. G., . . . Zhani, M. F. (2013). Data center network virtualization: A survey. Communications Surveys & Tutorials, IEEE, 15(2), 909-928.

Beheshti, N., & Zhang, Y. (2012). Fast failover for control traffic in Software-defined Networks. Paper presented at the Global Communications Conference (GLOBECOM), 2012 IEEE.

Benson, T., Akella, A., Shaikh, A., & Sahu, S. (2011). CloudNaaS: a cloud networking platform for enterprise applications. Paper presented at the Proceedings of the 2nd ACM Symposium on Cloud Computing.

Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., . . . Snow, W. (2014). ONOS: towards an open, distributed SDN OS. Paper presented at the Proceedings of the third workshop on Hot topics in software defined networking.

Blenk, A., & Kellerer, W. (2013). Traffic pattern based virtual network embedding. Paper presented at the Proceedings of the 2013 workshop on Student workshop.

Calyam, P., Rajagopalan, S., Selvadhurai, A., Mohan, S., Venkataraman, A., Berryman, A., & Ramnath, R. (2013). Leveraging OpenFlow for resource placement of virtual desktop cloud applications. Paper presented at the Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on.

Casado, M. (2013). OpenStack and Network Virtualization. Retrieved 2013, April 17, from http://blogs.vmware.com/tribalknowledge/2013/04/openstack-and-network-virtualization.html

VOL. 10, NO. 3, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

Daniel Philip, V., & Gourhant, Y. (2014). Cross-control: A scalable multi-topology fault restoration mechanism using logically centralized controllers. Paper presented at the High Performance Switching and Routing (HPSR), 2014 IEEE 15th International Conference on.

Foster, N., Guha, A., Reitblatt, M., Story, A., Freedman, M. J., Katta, N. P., Schlesinger, C. (2013). Languages for software-defined networks. Communications Magazine, IEEE, 51(2), 128-134.

Gurbani, V. K., Scharf, M., Lakshman, T., Hilt, V., & Marocco, E. (2012). Abstracting network state in Software Defined Networks (SDN) for rendezvous services. Paper presented at the Communications (ICC), 2012 IEEE International Conference on.

Handigol, N., Seetharaman, S., Flajslik, M., Gember, A., McKeown, N., Parulkar, G., Krishnamurthy, A. (2011). Aster* x: Load-Balancing Web Traffic over Wide-Area Networks.

Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N., & Johari, R. (2009). Plug-n-Serve: Load-balancing web traffic using OpenFlow. ACM SIGCOMM Demo.

Hassas Yeganeh, S., & Ganjali, Y. (2012). Kandoo: a framework for efficient and scalable offloading of control applications. Paper presented at the Proceedings of the first workshop on Hot topics in software defined networks.

Ishimori, A., Farias, F., Cerqueira, E., & Abelém, A. (2013). Control of Multiple Packet Schedulers for Improving QoS on OpenFlow/SDN Networking. Paper presented at the Software Defined Networks (EWSDN), 2013 Second European Workshop on.

Jarschel, M., Wamser, F., Hohn, T., Zinner, T., & Tran-Gia, P. (2013). Sdn-based application-aware networking on the example of youtube video streaming. Paper presented at the Software Defined Networks (EWSDN), 2013 Second European Workshop on.

Kannan, K., & Banerjee, S. (2013). Compact TCAM: Flow Entry Compaction in TCAM for Power Aware SDN Distributed Computing and Networking (pp. 439-444): Springer.
Kant, K. (2009). Data center evolution: A tutorial on state of the art, issues, and challenges. Computer Networks, 53(17), 2939-2965.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Hama, T. (2010). Onix: A Distributed Control Platform for Large-scale Production Networks. Paper presented at the OSDI.

Kreutz, D., Ramos, F., & Verissimo, P. (2013). Towards secure and dependable software-defined networks. Paper presented at the Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking.

Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-Defined Networking: A Comprehensive Survey. arXiv preprint arXiv:1406.0440.

Levin, D., Wundsam, A., Heller, B., Handigol, N., & Feldmann, A. (2012). Logically centralized?: state distribution trade-offs in software defined networks. Paper presented at the Proceedings of the first workshop on Hot topics in software defined networks.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Turner, J. (2008). OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69-74.

Mendonca, M., Nunes, B. A. A., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2013). A Survey of software-defined networking: past, present, and future of programmable networks. hal-00825087.

Paul, S., & Jain, R. (2012). OpenADN: Mobile apps on global clouds using OpenFlow and Software Defined Networking. Paper presented at the Globecom Workshops (GC Wkshps), 2012 IEEE.

Phemius, K., Bouet, M., & Leguay, J. (2013). DISCO: Distributed multi-domain SDN controllers. arXiv preprint arXiv:1308.6138.

Raghavendra, R., Lobo, J., & Lee, K.-W. (2012). Dynamic graph query primitives for sdn-based cloudnetwork management. Paper presented at the Proceedings of the first workshop on hot topics in software defined networks.

Schmid, S., & Suomela, J. (2013). Exploiting locality in distributed sdn control. Paper presented at the Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking.

Shin, M.-K., Nam, K.-H., & Kim, H.-J. (2012). Software-defined networking (SDN): A reference architecture and open APIs. Paper presented at the ICT Convergence (ICTC), 2012 International Conference on.

Stallings, W. (2013). Software-defined networks and openflow. Internet Protocol J.

Stephens, B., Cox, A., Felter, W., Dixon, C., & Carter, J. (2012). PAST: Scalable Ethernet for data centers. Paper presented at the Proceedings of the 8th international conference on Emerging networking experiments and technologies.
Tootoonchian, A., & Ganjali, Y. (2010). HyperFlow: A distributed control plane for OpenFlow. Paper presented at

the Proceedings of the 2010 internet network management conference on Research on enterprise networking.

Wang, R., Butnariu, D., & Rexford, J. (2011). OpenFlow-based server load balancing gone wild: Hot-ICE.

Wasserman, M., & Hartman, S. (2013). Security analysis of the open networking foundation (onf) openflow switch specification.

Yin, H., Xie, H., Tsou, T., Lopez, D., Aranda, P., & Sidi, R. (2012). SDNi: A message exchange protocol for software defined networks (SDNs) across multiple domains. submitted to IETF Internet-draft, December.